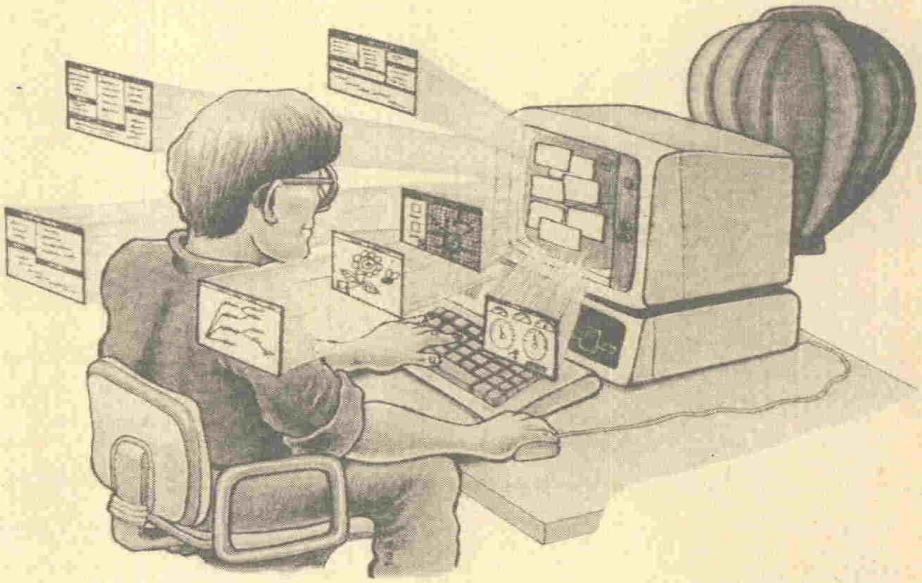
จเร เลิศสุดวิชัย

SOFT MAIL

พัฒนาโปรแกรมยูทิลิตี้ด้วย เทอร์โบซี

ตอนที่ 3



นำเสนอรายละเอียดของโปรแกรมยูทิลิตี้ที่น่าสนใจ อีกเช่นเคย

สำหรับในฉบับนี้มุมภาษาซีก็ยังจะขอเสนอโปรแกรมยูทิลิตี้เพื่อช่วยงานอีก 3 โปรแกรม คือ โปรแกรม RDIR สำหรับใช้เปลี่ยนชื่อไดเรคทอรีย่อย โปรแกรม VT สำหรับสร้างต้นไม้ซึ่งเป็นโครงสร้างการจัดเก็บไดเรคทอรีย่อยในดิสค์ และโปรแกรม DDEL สำหรับช่วยลบไฟล์ในไดเรคทอรีย่อยและทุก ๆ ไดเรคทอรีย่อยที่อยู่ในพาย โดยมีรายละเอียดดังนี้ครับ

โปรแกรม RDIR

Rename Directory

การทำงาน

ในการใช้งานดอสมีบ่อยครั้งที่เรามีความต้องการที่จะเปลี่ยนชื่อไดเรคทอรีย่อย ซึ่งก็จะพบว่าดอสไม่ได้เตรียมคำสั่งหรือโปรแกรมที่ใช้งานนี้มาให้ บางท่านอาจคิดถึงคำสั่ง REN แต่คำสั่งนี้จะใช้เปลี่ยนเฉพาะชื่อไฟล์เท่านั้น ไม่สามารถใช้กับไดเรคทอรีย่อยได้ และบางท่านอาจใช้วิธีการง่าย ๆ โดยการคัดลอกไฟล์ในไดเรคทอรีย่อยที่ต้องการจะเปลี่ยน นำไปเก็บไว้ในไดเรคทอรีย่อยใหม่ที่สร้างเตรียมไว้ หลังจากนั้นจึงลบไฟล์ในไดเรคทอรีเดิมทิ้งพร้อมทั้งทำลายไดเรคทอรีทิ้งด้วย ซึ่งวิธีนี้ไม่ใช่วิธีการที่ดี เพราะต้องมีการทำงานถึงสองขั้นตอน ทำให้ต้องเสียเวลาในการทำงาน และอาจจะเกิดปัญหาขึ้นทำให้งานไม่สำเร็จ เพราะเนื้อที่บนดิสค์มีไม่เพียงพอในการคัดลอกไฟล์

โปรแกรม RDIR จะเป็นโปรแกรมที่ช่วยงานดังกล่าว โดยวิธีการทำงานจะใช้ฟังก์ชันในไลบรารีของเทอร์โบซีในเปลี่ยนชื่อไฟล์ ซึ่งจะทำให้การทำงานมีความรวดเร็วและมีประสิทธิภาพ

การใช้งาน

```
C: \ > rdir dirname newname
```

โดยที่ dirname คือชื่อไดเรคทอรีที่ต้องการเปลี่ยนชื่อ และ newname คือชื่อไดเรคทอรีใหม่

ตัวอย่าง

```
C: \ > rdir d: \ fox d: \ foxbase
```

ตัวอย่างนี้เป็นการเปลี่ยนชื่อไดเรคทอรี fox ในไดรฟ์ D: ให้เป็นชื่อ foxbase

```
C: \ >rdir d: \ foxbase \ dbf
d: \ foxbase \ data
```

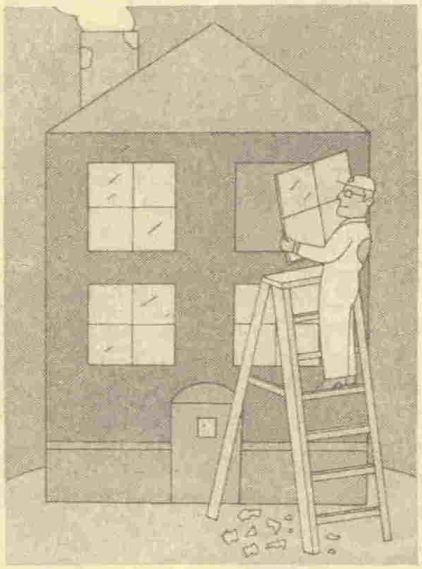
ตัวอย่างนี้เป็นการเปลี่ยนชื่อไดเรคทอรี dbf ซึ่งอยู่ภายใต้ไดเรคทอรี foxbase ในไดรฟ์ D: ให้เป็นชื่อ data

```
C: \ >rdir Lotus 123
```

ตัวอย่างนี้เป็นการเปลี่ยนชื่อไดเรคทอรี Lotus บนไดรฟ์ปัจจุบัน (คือไดรฟ์ C:) ให้เป็นชื่อ 123

เทคนิคโปรแกรม

โปรแกรม RDIR ใช้ฟังก์ชันไลบรารีของเทอร์โบซีคือฟังก์ชัน rename ( ) ในการเปลี่ยนชื่อหรือย้ายไฟล์จากไดเรคทอรีหนึ่งไปยังอีกไดเรคทอรีหนึ่ง (ซึ่งในโปรแกรม RM ที่ใช้ย้ายไฟล์ในมุมภาษาซีฉบับที่แล้วก็ใช้ฟังก์ชันนี้เช่นกัน) และข้อจำกัดของโปรแกรมคือ



ไดเรคทอรีที่ต้องการเปลี่ยนจะต้องอยู่บนไดรฟ์เดียวกันเท่านั้น ซึ่งข้อจำกัดเกิดขึ้นจากตัวฟังก์ชัน rename ( ) นั่นเอง การใช้งานฟังก์ชัน rename( ) มีรูปแบบดังนี้

```
int rename (char *oldname,
char *newfname);
```

การทำงานนี้จะให้ค่าเป็นศูนย์หากการเปลี่ยนชื่อสำเร็จ และหากการเปลี่ยนชื่อไม่สำเร็จก็จะให้ค่าลบ ซึ่งข้อผิดพลาดของการทำงานจะบอกไว้ในตัวแปรระบบของเทอร์โบซีชื่อ errno โดยค่าใน errno จะบอกถึงความผิดพลาด ถ้าค่าในตัวแปรมีค่าดังต่อไปนี้ (ดูในไฟล์ Stdio.h)

– ENOENT ไม่มีไฟล์หรือไดเรคทอรี

– EACCES ไม่มีสิทธิ์เปลี่ยนชื่อเนื่องจากแอตทริบิวต์ของไฟล์ไดเรคทอรีหรือมีชื่อไฟล์/ไดเรคทอรีอยู่ออกแล้ว

– ENOTSAM ระบุไดรฟ์ไม่ถูกต้อง

โปรแกรม VT

Visual Tree

การทำงาน

โปรแกรมจะทำงานโดยอ่านเอาโครงสร้างของไดเรคทอรีที่จัดเก็บในฮาร์ดดิสค์หรือฟลอปปี้ดิสค์มาแสดงรูปของต้นไม้ (Tree) จะคล้ายกับคำสั่ง Tree ของดอส แต่การแสดงผลของคำสั่ง Tree จะแสดงเพียงชื่อไดเรคทอรีและไดเรคทอรีย่อยที่อยู่ภายในก็จะแสดงเพียงชื่อเช่นกัน แต่ย่อยหน้าให้ทราบว่าอยู่ในไดเรคทอรีอะไรเท่านั้น ซึ่งลักษณะโครงสร้างต้นไม้ที่ได้จากโปรแกรม VT จะช่วยให้เข้าใจโครงสร้างไดเรคทอรีได้ดีกว่า

การใช้งาน

```
C: \ > vt
```

โปรแกรม VT ไม่ต้องการพารามิเตอร์อะไร โดยโปรแกรมจะอ่านโครงสร้างไดเรคทอรีจากไดรฟ์ปัจจุบันมาทำงาน

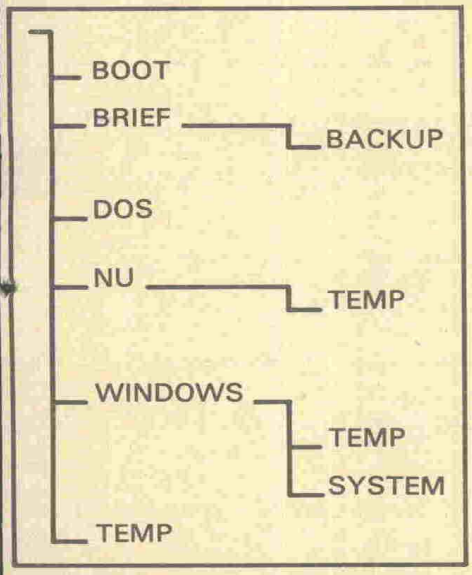
ตัวอย่าง

```
C: \ vt
```

โครงสร้างการจัดเก็บไฟล์ในไดรฟ์ C: จะถูกแสดงดังรูปที่ 1

เทคนิคโปรแกรม

โปรแกรม VT มีการทำงานเช่นเดียวกับโปรแกรมยูทิลิตี้อื่นที่เคยเสนอไปแล้ว ฟังก์ชันที่ใช้งานก็คือ findfirst และ findnext โดยจะเริ่มต้นจากไดเรคทอรีรากและเข้าค้นในไดเรคทอรี



รูปที่ 1 แสดงโครงสร้างการจัดเก็บไดร์ฟ C:

ย่อยอื่น ๆ เก็บโครงสร้างเอาไว้ในตัวแปร temp จากนั้นจะถูกสร้างให้เป็นโครงสร้างต้นไม้ด้วยฟังก์ชัน displaytree

โปรแกรม DDEL

Directory-Delete

การทำงาน

โปรแกรม DDEL มีประโยชน์มากในกรณีที่ต้องจะลบหรือล้างไดเรคทอรีทิ้ง โดยที่ภายในไดเรคทอรีมีไดเรคทอรีย่อยเติมไป ซึ่งตามปกติการลบไดเรคทอรีโดยใช้คำสั่ง rd จะทำได้ก็ต่อเมื่อในไดเรคทอรีนั้นว่างอยู่ ดังนั้นการล้างไดเรคทอรีจะต้องทำถึง 2 ขั้นตอนด้วยกันคือ การเข้าไปในไดเรคทอรีย่อยต่างของไดเรคทอรีที่ต้องการลบ แล้วลบไฟล์ในนั้นทิ้ง จากนั้นจึงย้ายออกจากไดเรคทอรีย่อย ใช้คำสั่ง rd เพื่อล้างไดเรคทอรีย่อยนั้นทิ้ง แล้วค่อย ๆ ทำกับทุกไดเรคทอรีย่อยที่มีอยู่จนกระทั่งในไดเรคทอรีที่ต้องการลบทิ้งว่างลง จึงจะใช้คำสั่ง rd ล้างไดเรคทอรีดังกล่าวได้

โปรแกรม DDEL จะช่วยในการทำงานดังกล่าวให้ ซึ่งเมื่อสั่งงานโปรแกรม DDEL ต้องระบุไดเรคทอรีที่ต้องการล้าง โปรแกรม DDEL จะย้ายเข้าไปในไดเรคทอรีที่ระบุและตรวจสอบไดเรคทอรีย่อยให้เอง และจัดการล้างไดเรคทอรีย่อยต่าง ๆ ภายในไดเรคทอรีที่ระบุจนหมด จากนั้นจะล้างไดเรคทอรีที่ระบุให้ในขั้นตอนสุดท้าย

ตัวอย่าง

```
C: \ >ddel d: \ clp501
```

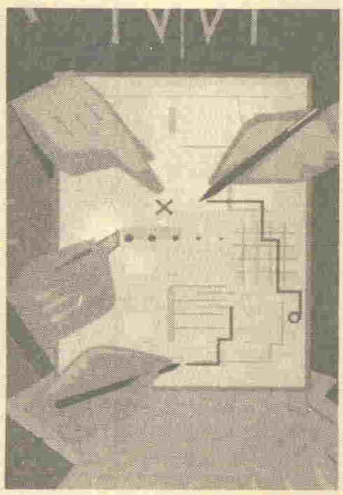
เป็นการลบไดเรคทอรีคลิปเปอร์ 5.01 ในไดรฟ์ D: เพื่อจะลงซอฟต์แวร์รุ่นใหม่ลงไป ไฟล์ทั้งหมดภายใต้ไดเรคทอรี clp501 จะถูกล้างทิ้งหมด

```
C: \ >ddel \ lotus
```

ลบไดเรคทอรี lotus ในไดรฟ์ C: ซึ่งเป็นไดเรคทอรีใช้งานปัจจุบัน

```
C: \ TC \ >ddel d: \
```

จะเป็นการลบไฟล์ในไดรฟ์ D: ทุกไฟล์และทุกไดเรคทอรีทั้งหมด จุดที่เป็นข้อควรระมัดระวังของการใช้งานโปรแกรม DDEL ซึ่งถ้าหากเผลอไปไฟล์ทั้งหมดจะถูกลบทิ้งหมด อย่างไรก็ตาม โปรแกรมจะเตือนและถามความแน่ใจอีกครั้งว่าต้องการจะลบไฟล์ทั้งไดรฟ์ทิ้งจริงหรือไม่



```
C: \ > ddel
```

การเรียกชื่อโปรแกรมอย่างเดียวโปรแกรมจะแสดงวิธีการใช้ให้

เทคนิคโปรแกรม

เทคนิคของโปรแกรม DDEL ก็จะเป็นเทคนิคเดียวกันกับที่ใช้ในโปรแกรมยูทิลิตี้ที่เกี่ยวกับไดเรคทอรีที่ผ่าน ๆ มาแล้ว จะต่างกันก็แต่เพียงเมื่อโปรแกรมพบไดเรคทอรีจะย้ายเข้าไปข้างในและทำการลบไฟล์ทิ้งหมดแล้วถอยกลับมายังไดเรคทอรีเดิม และทำต่อในลักษณะเดิมจนกระทั่งหมดทั้งไดเรคทอรี

โปรแกรมจะตรวจสอบด้วยว่าหากมีการระบุการลบไฟล์ที่ไดเรคทอรีราก โปรแกรมจะเตือนก่อนและถามความแน่ใจอีกครั้ง ซึ่งหากผู้ใช้แน่ใจและตอบ Yes โปรแกรมจะลบไฟล์ทั้งหมดในดิสก์ทิ้งทันที

ฟังก์ชันที่ใช้งานในโปรแกรมนี้ก็คือ findfirst( )/findnext( ) และในฟังก์ชัน deldir( ) เป็นฟังก์ชันที่มีการเรียกตัวเองหรือรีเคอชัน (Recursion)

โปรแกรมต้นฉบับ

เช่นเดิมครับ โปรแกรมต้นฉบับของ RDIR.C, VT.C และ DDEL.C จะอยู่ในรูปที่ 2, 3 และ 4 ตามลำดับ และหากผู้ที่ต้องการโปรแกรมก็หาได้จากซอฟต์เมล์ประจำฉบับครับ

รูปที่ 2 โปรแกรม RDIR.C

```
 1 /*
 2
 3   Program          : RENDIR.C
 4
 5   Purpose          : RENDIR performs a task which DOS desn't. Renaming
 6                        directory can be necessary when upgrading sofware
 7                        on you hard disk or even when you are just doing
 8                        some disk housekeeping. The program will also
 9                        rename files, but since DOS provides this feature
10                        there is not reason to use this utility for that
11    Requirements : Program will work on MS-DOS and PC-DOS
12                        computer using version 2.0 or greater.
13
14
15 */

17 #include <stdio.h>
18 #include <dir.h>
19 #include <string.h>

21 main (int argc, char *argv[])
22 {
23   /* variables used in the main program. */
24   char presname[MAXPATH], futname[MAXPATH];

26   printf ("\nRENAME-Rename Dir, Micro-computer.\n");

28   if (argc > 2)
29     {
30     strcpy (presname,argv[1]);
31     strcpy (futname,argv[2]);
32     }
33   else
34     {
35     printf ("\nUsage: C:\>rendir dirname newname\n");
36     return (1);
37     }
38   if (rename(presname,futname) == 0)
39     return (0);
40   else
41     {
42     printf("\nThere was a problem renaming the directory!\n");
43     return (2);
44     }
45 }
```

รูปที่ 3 โปรแกรม VT.C

```
 1 /*
 2
 3   Title              : VT.EXE   (Visual Tree)
 4   Purpose            : With this utility you can view a visual tree of
 5                          all the directores which you have access to. It
 6                          is like the LISTDIR utility but slightly different
 7                          because it gives you the graphic overview of your
 8                          system instead of the text vased output that the
 9                          other utility give you
10    Requirements     : The program will work on MS-DOS and PC-DOScomputer
11                          using version 2.0 or later
12
13 */

15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <string.h>
18 #include <dir.h>
19 #include <dos.h>

21 /* The two internal procedure used in this program. */

23 void findirs (void);
24 void displaytree (void);

26 /*
27 The array of line - appropriately called LINES - is used
28 for storing output that would normally  be printed on
29 the screen. The reason that this is done is so that the
30 procedure a pretier display.
31 */

33 char  temp[100], lines[320][100];
34 int   count, i, deep;

36 /*
37 The main() loop is used for initialization and for calling
38 the most important part of this utility, FINDIRS. Once
39 FINDIIRS has done its work and has created a preliminary
40 visual tree, the DISPLAYTREE procedure is called. It fixes
41 up any minor details that are necessary and then displays
42 the visual tree on the screen

44 CALLED FROM:   (DOS)
45 CALLS:        findirs    displaytree

47 */

49 main ()
50 {
51   /* Variable used in the main program */

53   char origdir[MAXPATH];

55   printf ("\nVT-Visual Tree, TC-Edition\n");

57   memset (&lines[0][0], ' ', sizeof (lines));

59   getcurdir (0, origdir);          /* Get current directory */
60   chdir ("\\");                    /* Switch to root */
61   deep = 0;
62   count = 0;
63   sprintf (&lines[count][0],"\\¿"); /* Two back slashes and ASCII 191 */
64   findirs ();
65   displaytree ();
66   chdir (origdir);                 /* Back to original */
67 }

69 /* The function findirs() searchs for all directores.
70 Hidden ones and system ones are also found. It stores
71 the information which it gathers into an array of lines.
72 Most of the formatting takes place in this procedure, but
73 is later enhanced by function displaytree.

75 CALLED FROM:   MAIN

77 CALLS:        (none)
```

ปที่ 3 (ต่อ)

```
 79 */

 81 void findirs (void)
 82 {
 83   /*  Variables used in this procedure  */

 85   int done;
 86   struct ffblk dta;

 88   deep++;
 89   done = findfirst ("*.*", &dta, FA_DIREC+FA_SYSTEM+FA_HIDDEN);

 91   while (!done)
 92     {
 93     if (((dta.ff_attrib & FA_DIREC) == FA_DIREC) &&
 94        (dta.ff_name[0] != '.'))
 95        {
 96        if (deep == 1)
 97          {
 98          temp [0] = 32;
 99          temp [1] = 195;
100           temp [2] = 0;
101          }
102        if (deep > 1)
103          {
104          for (i = 0; i <= (deep-1) * 14; i++)
105             temp [i] = 32;

107          temp [1] = 179;
108          temp [i] = 195;
109          temp [i+1] = 0;
110          }

112        strcat (temp, dta.ff_name);
113        count++;
114        sprintf (&lines [count][0], "%s", temp);

116        chdir (dta.ff_name);
117        findirs ();
118        deep--;
119        chdir ("..");
120        }
121      done = findnext (&dta);
122      }
123 }

125 /* The function displaytree() is used for displaying
126 the data which the previous procedure formats. It goes
127 through the array of lines and figures out where different
128 graphics charaters should be inserted to make the tree
129 more appealing.

131    All the strnge graphics characterswhich are used are
132 described with ASCII numerical equivalets within comments.
133 This should make it easy for you to type them in. To get
134 a character within the TC enviroment just hold down the ALT
135 key keypad. The strange symbol will be created for you.

137 CALLED FROM:    main

139 CALLS:         (none)
140 */
141 void displaytree (void)
142 {
143   /* Variables used in this procedure. */
144   int j, k, ok;

146   for (i = 0 ; i<= count;i++)
147     {
148     for (j = 0; j <= 99; j++)
149       {
150       ok = 0;

151       if ((lines [i][j] == 'Ã') &&      /* ASCII 195 */
152          (lines[i-1][j] == ' '))     /* and 32  */
153          {
154          strcpy (temp, &lines[i-1][0]);
155          for (k = strlen (temp); k < j; k++)
156            {
157            strcat (temp, "Ä");       /* ASCII 196 */
158            ok = 1;
159            }
160          if (ok)
161            strcat (temp,"¿");        /* ASCII 191 */
162          sprintf (&lines [i-1][0], "%s", temp);
163          }
164       if ((lines[i][j] == 'Ã') &&     /* ASCII 195 */
165          (lines [i+1][j] == ' '))     /* and 32  */
166          {
167          strcpy (temp,&lines[i][0]);
168          for (k = strlen (temp); k >= 0; k--)
169            if (temp [k] == 'Ã')      /* ASCII 195 */
170              temp [k] = 'À';         /* and 192 */
171          sprintf (&lines [i][0], "%s", temp);
172          }
173       if (( lines[i][j] == 'Ã') &&    /* ASCII 195 */
174          (lines [i-1][j] == ' '))     /* and 32  */
175          {
176          k = i - 1;
177          while (lines[k][j] != 'À')   /* ASCII 192 */
178            {
179            lines [k][j] = '³';       /* ASCII 179 */
180            k--;
181            }
182          lines [k][j] = 'Ã';          /* ASCII 195 */
183          }
184       }
185     }
186   i = count;                         /* ASCII 192 */
187   if (lines [i][1] != 'À')
188     {
189     while ((i >= 0) &&
190           (lines [i][1] != 'Ã'))     /* ASCII 195 */
191       {
192       if (lines[i][1] == '³')        /* ASCII 179 */
193         lines [i][1] = ' ';          /* and 32 */
194       i--;
195       }
196     lines[i][1] = 'À';               /* ASCII 192 */
197     }
198   printf ("\n");
199   for (i = 0;i <= count; i++)
200     {
201     printf ("%s\n", &lines[i][0]);
202     }
203 }
```

รูปที่ 4 โปรแกรม DDEL.C

```
/*

 Program          : DDEL.EXE (Directory Delete)

 Purpose          : Trashes an entire subdirectory structure. It will
                    erase every file in a subdirectory tree and then
                    remove the subdirectories as well. for example,
                    You can use DDEL to get rid of a directory and
                    every subdirectory under it in one step
 Requirements : Program will work on MS-DOS and PC-DOS
                    computer using version 2.0 or laster.


*/

#include <stdio.h>
#include <dir.h>
#include <string.h>
#include <conio.h>
#include <dos.h>
#include <ctype.h>
#include <process.h>

/*
The two function used in this program. The GETDIRECTORY
function returns a pointer to a text string.
*/

void deldir (void);
char *getdirectory (char *path);

/*
Global variables. The varible deep is used
for keeping track of how deep the program is
into a subdirectory
*/

char curdir[MAXPATH], newdir [MAXPATH],
    name [MAXPATH];
int  problem, i, deep;

/*
The main function of this program sets up various variables which
the program needs. It stores things like the current drive and
directory and current path of the drive which the user specifies
on the command line.

CALLED FROM:   (Dos)
CALLS:         deldir      getdirectory
*/

main (int argc,char *argv[])
{
  char origdir [MAXPATH], target [MAXPATH],
      trashdir [MAXPATH];

  printf ("\nDDEL-Directory Delete, TC Edition.\n");

  if (argc > 1)
    {
    strcpy (trashdir, argv[1]);
    i = strlen (trashdir);
    if (i < 3)
      {
      printf ("Too few characters for path!\n");
      return (1);
      }

    if ((trashdir[i-1] == 92) & (i != 3))
      trashdir[i-1] = 0;
    strupr (trashdir);
    if (i == 3)
      {
      printf ("\a\aYou specified \"%s.\" This is very dangerous! \a\n",
          trashdir);
      printf ("Press Y to delete \"%s.\" or any other key to quit. \n\n",
          trashdir);
      if (toupper(getch()) != 'Y')
        {
        printf ("\n");
        exit(2);
        }
      }
    }
  else
    {
    printf ("Usage:C:\>ddel path\n");
    printf ("Where path is like c:\\temp\n");
    return (3);
    }

  deep = 0;
  getdirectory(origdir);
  setdisk(trashdir[0]- 65);
  getdirectory(target);
  problem = chdir (trashdir);
  if (!problem)
    {
    deldir ();

    if ((strlen(target) == 2) && (target[1] == ':'))
      chdir("\\");
    else
      chdir(target);

    if (strcmp(trashdir,target) <= 0)
      {
      if (strlen (trashdir) > 3)
        {
        chdir ("..");
        rmdir(trashdir);
        }
      }
    else
      {
      rmdir (trashdir);
      }
    }
  else
    {
    printf ("The specified directory does not exist.\n");
    }

  setdisk (origdir[0] - 65);

  if ((strlen (origdir) == 2) && (origdir[1] == ':'))
    chdir ("\\");
  else
    chdir(origdir);

  return (0);
}

/*
This function searchs for directories. If it finds one, it switches
into it and calls itself recursively to find more directories. By
doing this, it find all directories under a given path. Once it can't
find more directories,it start deleting files and finally backtracks
and remove the directores one by one. It is a very dangerous function.

CALLED FROM:    main

CALLS:          getdirectory
*/

void deldir (void)
```

รูปที่ 4 (ต่อ)

```
147 {
148 /*
149 Only one variable and structure is stacked with recursive
150 calls to this procedure.
151 */
152   int done;

154   struct ffblk dta;

156   getdirectory (curdir);
157   deep++;

159   done = findfirst("*.*",&dta,FA_DIREC+FA_SYSTEM+FA_HIDDEN);

161   while (!done)
162   {
163   if (((dta.ff_attrib & FA_DIREC) == FA_DIREC) &&
164     (dta.ff_name[0] != '.'))
165     {
166     sprintf(newdir,"%s\\%s",curdir,dta.ff_name);
167     chdir(newdir);
168     deldir();
169     deep--;
170     chdir("..");
171     rmdir(curdir);
172     getdirectory(curdir);
173     }
174     done = findnext(&dta);
175   }

177   for (i = 0;i <= deep * 3; i++)
178     printf("-");

180   if (strlen(curdir) > 2)
181     printf ("%s\n\n",curdir);
182   else
183     printf ("%s\\\n\n",curdir);

185   done = findfirst("*.*",&dta,47);

187   while (!done)
188     {
189     for (i = 0; i <= deep * 3; i++)
190       printf (" ");
191     printf ("%s\n",dta.ff_name);
192     sprintf(name,"%s\\%s",curdir,dta.ff_name);
193     unlink(name);
194     done = findnext(&dta);
195     }
196   printf ("\n");
197 }

199 /*
200 This function create a string which holds the drive and path.
201 This drive number which GETDISK returns is added to the letter A
202 (the base disk drive) to get the letter of the current disk drive.
203 When the function is over, a pointer to the drive and path is
204 returned to the caller.
205 */

207 char *getdirectory(char *path)
208 {
209   strcpy (path, "X:\\");
210   path[0] = 'A' + getdisk();
211   getcurdir (0,path+3);

213   if (strlen(path) == 3)
214     path[2] = 0;
215   return (path);
216 }
```


พบแล้ว

สร้างภาพกราฟิก
และเอฟเฟ็กต์โดยเร็ว
การอัพเกรด CoreDRAW
จากเวอร์ชันก่อน ๆ

พลาดไม่ได้

หนังสือ เริ่มต้นใช้งาน Core DRAW 3 **เป็นหนังสือสองเล่มในเล่มเดียวกัน เป็นหนังสือแนะนำ** CorelDRAW **เวอร์ชันล่าสุด ประกอบด้วยตัวอย่างบทเรียนและหลักฐานอ้างอิงคำสั่งต่าง ๆ ที่ใช้ในโปรแกรมนี้อย่างครบถ้วนสมบูรณ์ ไม่ว่าท่านผู้อ่านจะยังใหม่ต่อโปรแกรม** Corel DRAW **หรือว่าต้องการจะอัพเกรด** CorelDRAW **จากเวอร์ชันก่อน ๆ ก็ตาม**

หาซื้อได้ตามร้านหนังสือชั้นนำทั่วไป
หรือที่ ซีเอ็ดบุ๊คเซ็นเตอร์ทุกสาขา
และสั่งซื้อได้ที่ซีเอ็ด สำนักงานใหญ่
โทร 248-6280

มีจำหน่ายแล้ว